AMENDMENTS TO THE CLAIMS

The listing of claims will replace all prior versions, and listings, of claims in the application:

1. (Currently Amended) A computer system for supporting a plurality of floating point architectures, each floating point architecture having at least one format, the system comprising:

a floating point unit having an internal data-flow according to an internal floating point format for performing floating point operations in the internal format, wherein the internal format has a number of exponent bits which is at least athe minimum number required to support each of the plurality of floating point architectures and the internal format has a number of fraction bits which is at least the minimum number required to support each of the plurality of floating point architectures; and

a converter for converting an exponent value corresponding to each one of the plurality of floating point architectures into the internal floating point format such that an operand of any one of the plurality of floating point architectures input to the floating point unit is converted into the internal floating point format for operation by the floating point unit, and the result of the operation is converted back into the one of the plurality of floating point architectures by converting an exponent value corresponding to the internal floating point format into the one of the plurality of floating point architectures, wherein said converting into the internal floating point format occurs without incurring additional clock cycles when said one of the plurality of floating point architectures is binary and said converting into the internal floating point format occurs without incurring additional clock cycles when said one of the plurality of floating point architectures is hexadecimal.

2. (Original) The computer system of claim 1, wherein the fraction bits corresponding to each of the plurality of floating point architectures are used by the floating point unit in an unconverted state.

3. (Original) The computer system of claim 1, wherein the plurality of floating point architectures include IBM®-S/390® hexadecimal floating point architecture and IEEE-754 binary floating point architecture.

4. (Original) The computer system of claim 3, wherein the converter:

determines a sign bit; and

normalizes a resulting binary floating point number according to at least one of IBM®-S/390® and IEEE-754 normalization modes.

5. (Original) The computer system of claim 3, wherein the internal floating point format is a binary format that has a 16 bit exponent biased by 32,768, a sign bit, and a 56 bit fraction.

6. (Currently Amended) The computer system of claim 1, wherein the plurality of floating point architectures includes a binary architected format and an hexadecimal architected format, and the internal format is a binary internal format.

7. (Original) The computer system of claim 6, wherein the binary internal format has a common predetermined fraction type corresponding to both of the hexadecimal and the binary architected formats.

8. (Original) The computer system of claim 7, wherein the numbers represented in the binary internal format corresponding to each of the hexadecimal architected format and the binary architected format, respectively, each have predetermined bias types that differ in the locations of the implied radix points.

9. (Original) The computer system of claim 8, wherein the binary internal format has a nonzero positive integer number M of exponent bits and a bias equal to $2^{(M-1)} - 1$, where M is the length of the internal exponent field.

10. (Original) The computer system of claim 1, wherein the plurality of floating point architectures includes a binary architected format and a hexadecimal architected format, the internal format is a binary internal format, and the converter comprises:

a first converter portion that converts the hexadecimal architected format to the binary internal format, and converts the binary architected format to the binary internal format; and

a second converter portion that converts the binary internal format into the binary architected format, and converts the binary internal format into the hexadecimal architected format.

11. (Original) The computer system of claim 10, wherein the first converter portion comprises an input format conversion multiplexor and input format conversion control, and the second converter portion comprises an output format conversion multiplexor and output format conversion control.

Q1

12. (Currently Amended) A floating point unit for supporting a first floating point architecture and a second floating point architecture, the first and second floating point architectures each having at least one format, the unit comprising:

an internal floating point format compatible with both the first and second floating point architectures, and sharing the fraction bits and sign bit with both of the floating point architectures;

a first converter to convert an operand of the first floating point architecture type to the internal floating point format by multiplexing the exponent bits of the operand, and to convert an operand of the second floating point architecture type to the internal floating point format by multiplexing the exponent bits of the operand, wherein said first converter operates without incurring additional clock cycles when said first floating point architecture type is binary, said first converter operates without incurring additional clock cycles when said first floating point architecture type is hexadecimal, said first converter operates without incurring additional clock cycles when said second floating point architecture type is binary, and said first converter

operates without incurring additional clock cycles when said second floating point architecture type is hexadecimal;

a floating point unit having an internal data-flow for the internal floating point format that supports the converted data of both the first and second floating point architectures, and that performs floating point operations on data formatted in the internal floating point format by the first converter; and

a second converter to convert data of the internal floating point format into data of the ~~transformed~~-first floating point architecture by multiplexing the exponent bits of the operand, and to convert data of the internal floating point format into data of the ~~transformed~~-second floating point architecture by multiplexing the exponent bits of the operand.

13. (Canceled)

a1

14. (Currently Amended) The floating point unit of claim 12, wherein the data in the internal floating point format is directly converted into data of the first floating point architecture by the secondthird converter with no additional delay, and wherein data in the internal floating point format is directly converted into data of the second floating point architecture by the secondfourth converter with no additional delay.

15. (Original) The floating point unit of claim 12, wherein the internal floating point format has a minimum nonzero positive number of exponent bits, N, to support both first and second floating point architectures.

16. (Original) The floating point unit of claim 12, wherein the internal floating point format has a single fraction type corresponding to the fraction portions of both the first and second floating point architectures.

17. (Original) The floating point unit of claim 12, wherein the internal format has the same fraction type as both of the first and second floating point architectures.

18. (Currently Amended) A method for processing data corresponding to a plurality of floating point architectures, comprising:

determining a type of ~~the instant~~ a floating point architecture;

converting the exponent of the data corresponding to the determined architecture into an internal floating point format by:

correcting the radix point location of the exponent data, and

pre-aligning the corrected exponent data, wherein said converting is performed without incurring additional clock cycles when said determined architecture is binary and said converting is performed without incurring additional clock cycles when said determined architecture is hexadecimal;

a1

performing arithmetic computations on the converted data;

re-converting the exponent of the computed data by:

re-correcting the radix point location of the exponent of the computed data, and

post-aligning the re-corrected exponent of the computed data; and

outputting the re-converted data to thereby provide floating point data corresponding to the original floating point architecture without incurring a delay beyond that required to compute the resulting output in the internal floating point format.